User Interface - Bug #2854

Bug # 2677 (New): fix drawing and functional differences between P2J GUI and 4GL GUI

scrollbar buttons don't draw the pressed state

11/16/2015 03:27 PM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Eugenie Lyzenko	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 11/16/2015 03:28 PM - Greg Shah

In the Swing client, ./combo_box/combo_box9_1.p can be used to see the problem. This is only a button drawing issue. The drop-down box does scroll in response to the button presses. The issue is that the scrollbar buttons don't animate their drawing of a pressed and then released button. The button always draws in the released state.

#2 - 11/16/2015 03:38 PM - Hynek Cihlar

There is another problem with GUI button pressed state and possibly related. When button is pressed in ALERT-BOX (and possibly elsewhere) and mouse is moved while holding button down, the button down draw state changes to button up. Tested in Swing GUI task branch 2677a.

#3 - 11/24/2015 09:32 AM - Greg Shah

- Assignee set to Eugenie Lyzenko

#4 - 11/24/2015 06:15 PM - Eugenie Lyzenko

- File evl_upd20151124b.diff added

The changes for review to fix. We need to use repaint() for pressing state. In drop-down based combo-box the special event processing loop stops proper scroll bar screen synchronization. Also is this change I have fixed the pressed state drawing for drop-down button in combo-box widget itself.

#5 - 11/24/2015 06:28 PM - Eugenie Lyzenko

The point #2 is still under resolution. Be ready later today.

#6 - 11/24/2015 08:09 PM - Eugenie Lyzenko

- File evl_upd20151124c.diff added

This is the change for review to note #2 resolution.

I have found another deviation with 4GL. When press and hold the mouse key on SB button the scrolling is turned to auto-scroll mode, content receive the scroll events permanently until the mouse become released. In P2J it is not implemented. Do we need to have this feature now?

#7 - 11/24/2015 09:45 PM - Eugenie Lyzenko

- File evl_upd20151124d.diff added

This is implementation for auto-scroll mode as reaction to press and hold the mouse on scrollbar button.

#8 - 11/25/2015 07:32 AM - Greg Shah

Code Review evl_upd20151124d.diff

My primary concern is whether the new approach has any negative ramifications for the web client. The web client's mouse event support is quite tricky and we have been trying to make it more independent of the Java side. We go to great lengths to process as much as possible on the javascript side.

Constantin: please review.

#9 - 11/25/2015 08:22 AM - Constantin Asofiei

Greg Shah wrote:

> Code Review evl_upd20151124d.diff
>
> My primary concern is whether the new approach has any negative ramifications for the web client. The web client's mouse event support is quite tricky and we have been trying to make it more independent of the Java side. We go to great lengths to process as much as possible on the javascript side.
>
> Constantin: please review.

Greg, I don't think this logic can be duplicated on the JS-side yet (as it is pretty complex). The code looks OK, but:

1. in the Web client, there is a small flash (on the scrollbar, I guess from redrawing) on initial button down of a drop-down's scroll button; otherwise the Web client responds OK.
2. the ScrollBarGuiButtonImpl.up field needs to be made volatile
3. both ScrollBarGuiImpl.mouseInputExecutor and ScrollBarGuiButton.mouseInputExecutor need to be stopped, when the widget is destroyed (otherwise there will be a thread leak).

#10 - 11/25/2015 08:43 AM - Greg Shah

Eugenie: please resolve items 2 and 3 from Constantin's feedback.

I'll add a task for the web client flashing issue.

#11 - 11/25/2015 09:04 AM - Hynek Cihlar

Eugenie Lyzenko wrote:

> This is implementation for auto-scroll mode as reaction to press and hold the mouse on scrollbar button.

Btw., why is the extra thread needed? Can't we just redraw the button back up on a mouse button released event?

#12 - 11/25/2015 09:13 AM - Hynek Cihlar

Hynek Cihlar wrote:

> Eugenie Lyzenko wrote:
>
> > This is implementation for auto-scroll mode as reaction to press and hold the mouse on scrollbar button.
>
> Btw., why is the extra thread needed? Can't we just redraw the button back up on a mouse button released event?

I see there is the repeated-clicks going on.

In any case, the event click should be dispatched on the main thread, otherwise this is a potential race condition. Also any thread-local storage would not work. See how this is done in ScrollBarGuiImpl.mousePressed().

#13 - 11/25/2015 09:17 AM - Eugenie Lyzenko

- File evl_upd20151125a.diff added

Greg Shah wrote:

> Eugenie: please resolve items 2 and 3 from Constantin's feedback.

Done with evl_upd20151125a.diff, please review.

#14 - 11/25/2015 09:23 AM - Eugenie Lyzenko

> Btw., why is the extra thread needed? Can't we just redraw the button back up on a mouse button released event?

No we can't. Because mouse remains pressed during this process. No mouse release event.

> I see there is the repeated-clicks going on.

This is not repeating clicks. This is repeating reaction to click. There is a difference.

#15 - 11/25/2015 11:59 AM - Greg Shah

Code Review evl_upd20151125a.diff

I'm OK with the changes.

Hynek/Constantin: do you have any remaining concerns?

#16 - 11/25/2015 12:07 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Greg Shah wrote:
>
> > Eugenie: please resolve items 2 and 3 from Constantin's feedback.
>
> Done with evl_upd20151125a.diff, please review.

You've made the ScrollBarGuiButton.mouseInputExecutor static ... I don't think you wanted to remain so, as you added code in destroy() to shutdown the executor.

#17 - 11/25/2015 12:22 PM - Hynek Cihlar

Greg Shah wrote:

> Code Review evl_upd20151125a.diff
>
> I'm OK with the changes.

> Hynek/Constantin: do you have any remaining concerns?

My remaining concern is the threading issue - the call to independentEventDrawingBracket() should go through the event queue, it must not be executed directly on the worker thread.

Eugenie, I think you can use ThinClient.invokeLater() for that or just create and post a mouse event.

#18 - 11/25/2015 04:19 PM - Eugenie Lyzenko

- File evl_upd20151125b.diff added

> You've made the ScrollBarGuiButton.mouseInputExecutor static ... I don't think you wanted to remain so, as you added code in destroy() to shutdown the executor.

Agree, it should not be static if we want to stop it. Changed.

> My remaining concern is the threading issue - the call to independentEventDrawingBracket() should go through the event queue, it must not be executed directly on the worker thread.
>
> I think you can use ThinClient.invokeLater() for that or just create and post a mouse event.

The auto scroll should work only when mouse is down. We need to react immediately. And this is safe because while mouse is down we can not do anything by mouse.

#19 - 11/25/2015 04:54 PM - Hynek Cihlar

Eugenie Lyzenko wrote:

> The auto scroll should work only when mouse is down. We need to react immediately. And this is safe because while mouse is down we can not do anything by mouse.

I don't agree. Other events are processed when the mouse button is down. You can move the mouse, press other mouse buttons, use the keyboard,

there may be server events coming in, etc. Even the logic executed by the worker thread itself posts paint events to the event queue. All the generated events are being dispatched on the main thread simultaneously with the worker thread logic and both may access the same unprotected resources.

#20 - 11/25/2015 05:01 PM - Eugenie Lyzenko

Hynek Cihlar wrote:

> Eugenie Lyzenko wrote:
>
> I don't agree. Other events are processed when the mouse button is down. You can move the mouse, press other mouse buttons, use the keyboard, there may be server events coming in, etc. Even the logic executed by the worker thread itself posts paint events to the event queue. All the generated events are being dispatched on the main thread simultaneously with the worker thread logic and both may access the same unprotected resources.

But not for the container we are scrolling in such a way. And for the time we are scrolling.

#21 - 11/25/2015 05:07 PM - Hynek Cihlar

Eugenie Lyzenko wrote:

> Hynek Cihlar wrote:
>
> > Eugenie Lyzenko wrote:
> >
> > I don't agree. Other events are processed when the mouse button is down. You can move the mouse, press other mouse buttons, use the keyboard, there may be server events coming in, etc. Even the logic executed by the worker thread itself posts paint events to the event queue. All the generated events are being dispatched on the main thread simultaneously with the worker thread logic and both may access the same unprotected resources.
>
> But not for the container we are scrolling in such a way.

The unprotected resources accessed by both threads are not limited to a container.

> And for the time we are scrolling.

No. During the time you are scrolling, the application is still processing input.

And even if you just click on the button you also get the race condition. There is no guarantee the two threads (main and the worker thread in the executor) won't access the same piece of data at the same time.

#22 - 11/25/2015 06:06 PM - Eugenie Lyzenko

- File evl_upd20151125c.diff added

Hynek Cihlar wrote:

> No. During the time you are scrolling, the application is still processing input.
>
> And even if you just click on the button you also get the race condition. There is no guarantee the two threads (main and the worker thread in the executor) won't access the same piece of data at the same time.

OK. Here is the new implementation.

#23 - 11/26/2015 02:46 AM - Hynek Cihlar

Eugenie Lyzenko wrote:

> Hynek Cihlar wrote:
>
> > No. During the time you are scrolling, the application is still processing input.
> >
> > And even if you just click on the button you also get the race condition. There is no guarantee the two threads (main and the worker thread in the executor) won't access the same piece of data at the same time.
>
> OK. Here is the new implementation.

Very well.

We also need to get rid of the ThinClient.eventDrawingBracket() call, the result of ThinClient.getInstance() is also a shared piece of memory that must not be accessed by two threads simultaneously and we want to also avoid processing the posted event on the worker thread. When you remove ThinClient.eventDrawingBracket() the posted event will be processed by the event loop on the main thread, which is what we desire.

#24 - 11/26/2015 03:34 AM - Eugenie Lyzenko

Hynek Cihlar wrote:

> Eugenie Lyzenko wrote:
>
> Very well.
>
> We also need to get rid of the ThinClient.eventDrawingBracket() call, the result of ThinClient.getInstance() is also a shared piece of memory that must not be accessed by two threads simultaneously and we want to also avoid processing the posted event on the worker thread. When you remove ThinClient.eventDrawingBracket() the posted event will be processed by the event loop on the main thread, which is what we desire.

This will cause the scrolling will not be visible. So this is the key point. The posting event approach ensures the events will be handled sequentially in addition to all other events are possibly coming.

#25 - 11/26/2015 03:43 AM - Hynek Cihlar

Eugenie Lyzenko wrote:

> Hynek Cihlar wrote:
>
> > Eugenie Lyzenko wrote:
> >
> > Very well.
> >
> > We also need to get rid of the ThinClient.eventDrawingBracket() call, the result of ThinClient.getInstance() is also a shared piece of memory that must not be accessed by two threads simultaneously and we want to also avoid processing the posted event on the worker thread. When you remove ThinClient.eventDrawingBracket() the posted event will be processed by the event loop on the main thread, which is what we desire.
>
> This will cause the scrolling will not be visible.

Do you mean the scroll bar is not redrawn? Is the scroll event properly processed, is the scroll bar state updated?

#26 - 11/26/2015 08:08 AM - Eugenie Lyzenko

> Do you mean the scroll bar is not redrawn? Is the scroll event properly processed, is the scroll bar state updated?

I mean neither scrollbar state itself is changing nor scrollable panel is scrolling while mouse is pressing. Until the button is releasing. Then all become properly updated. But this behavior is not correct and confuses the user. We need scrolling while mouse is pressing, not after.

#27 - 11/26/2015 08:43 AM - Hynek Cihlar

Eugenie Lyzenko wrote:

> > Do you mean the scroll bar is not redrawn? Is the scroll event properly processed, is the scroll bar state updated?
>
> I mean neither scrollbar state itself is changing nor scrollable panel is scrolling while mouse is pressing. Until the button is releasing. Then all become properly updated. But this behavior is not correct and confuses the user. We need scrolling while mouse is pressing, not after.

You have to use ThinClient.postOSEvent() so that the event manager is notified about the posted events.

#28 - 11/26/2015 09:37 AM - Eugenie Lyzenko

Hynek Cihlar wrote:

> Eugenie Lyzenko wrote:
>
> You have to use ThinClient.postOSEvent() so that the event manager is notified about the posted events.

This does not work as well.

#29 - 11/26/2015 11:08 AM - Hynek Cihlar

Eugenie Lyzenko wrote:

> Hynek Cihlar wrote:
>
> > Eugenie Lyzenko wrote:
> >
> > You have to use ThinClient.postOSEvent() so that the event manager is notified about the posted events.
>
> This does not work as well.

Interesting, this works for me fine. I have applied evl_upd20151125c.diff, removed eventDrawingBracket() and used ThinClient.postOSEvent() to post the scroll event. When I press left mouse button and hold the scroll bar is continuously scrolling.

Can you attach your diff?

#30 - 11/26/2015 12:13 PM - Hynek Cihlar

Btw., the same problem is in ScrollBarGuiImpl.mousePressed(). EventManager.postEvent() must be replaced with ThinClient.postOSEvent(). Eugenie, will you please update this as well?

#31 - 11/26/2015 02:36 PM - Eugenie Lyzenko

Hynek Cihlar wrote:

> Btw., the same problem is in ScrollBarGuiImpl.mousePressed(). EventManager.postEvent() must be replaced with ThinClient.postOSEvent(). Eugenie, will you please update this as well?

OK.

You are right, the ThinClient.postOSEvent() works fine. Previously I used EventManager.postOSEvent() and it was problematic.

After I'll refresh 2677b put new diff here.

#32 - 11/26/2015 03:11 PM - Eugenie Lyzenko

- File evl_upd20151126a.diff added

This is the updated diff for review.

#33 - 11/26/2015 03:25 PM - Hynek Cihlar

Eugenie Lyzenko wrote:

> This is the updated diff for review.

The threading issue is resolved.

Just one thing for future consideration. For UI "animations" (like auto scrolling) we should probably need a common executor managed by the framework, rather than each individual widget (and widget instance) managing its own.

#34 - 11/26/2015 03:34 PM - Eugenie Lyzenko

Hynek Cihlar wrote:

> Eugenie Lyzenko wrote:
>
> The threading issue is resolved.
>
> Just one thing for future consideration. For UI "animations" (like auto scrolling) we should probably need a common executor managed by the framework, rather than each individual widget (and widget instance) managing its own.

Greg, Hynek, Constantin,

So can I commit this into 2677b?

#35 - 11/27/2015 09:21 AM - Greg Shah

We know there are some regressions in the GUI testcases for 2677b. Let's see how significant these are before we decide if this will be checked in to 2677b.

#36 - 11/27/2015 01:06 PM - Greg Shah

Go ahead and check these into 2677b.

#37 - 11/27/2015 01:16 PM - Eugenie Lyzenko

The evl_upd20151125a.diff has been committed in 2677b as 10977.

#38 - 11/27/2015 02:10 PM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Closed

#39 - 11/16/2016 12:12 PM - Greg Shah

- Target version changed from Milestone 12 to GUI Support for a Complex ADM2 App

Files

evl_upd20151124b.diff	2.55 KB	11/24/2015	Eugenie Lyzenko
evl_upd20151124c.diff	4.92 KB	11/25/2015	Eugenie Lyzenko
evl_upd20151124d.diff	5.37 KB	11/25/2015	Eugenie Lyzenko
evl_upd20151125a.diff	7.23 KB	11/25/2015	Eugenie Lyzenko
evl_upd20151125b.diff	7.68 KB	11/25/2015	Eugenie Lyzenko
evl_upd20151125c.diff	8.59 KB	11/25/2015	Eugenie Lyzenko
evl_upd20151126a.diff	8.88 KB	11/26/2015	Eugenie Lyzenko